Homework 5

1. Stationarity properties. In this problem, we’ll explore the stationarity properties of the gradient method and the prox-gradient method.

(a) Show that, if $\theta^k = \theta^{k+1}$ for some $k$ in the gradient method, then $\theta^k$ is a stationary point. In other words, it satisfies
$$\nabla f(\theta^k) = 0.$$ 

(b) Show that, if $\theta^k = \theta^{k+1}$ for some $k$ in the gradient method, then $\theta^m = \theta^k$ for every $m > k$. In other words, $\theta^k$ is a fixed point of the method.

(c) What is the stopping condition for the prox-gradient method?

(d) Using the stopping condition from part (c), show that if $\theta^k = \theta^{k+1}$ for some $k$ in the prox-gradient method, then $\theta^k$ is a stationary point, so that it satisfies
$$\nabla f(\theta^{k+1}) = \frac{1}{h^k}(\theta^{k+1} - \theta^{k+1/2}).$$

*Hint.* Use the definition of $\theta^{k+1/2}$ given in the prox-gradient method slides and expand the right side of the equality above.

2. Introduction to Flux.jl. In the last homework, you learned how to implement basic loss functions and optimizers like a gradient descent. Having understood the mechanism behind their implementation, you will now use a package called Flux.jl, which implements them for you more efficiently. This question is your introduction to this package and its documentation. Going through the model zoo and using as many built-in methods is highly encouraged.

(a) You are given a dataset `flux.json` with features $x_i^T \in \mathbb{R}^d$ represented in a $n$-vector $x$ and labels $y_i \in \mathbb{R}$ represented in a $n$-vector $y$. Load the dataset stored and split it to 80-20 ratio.

(b) Train an affine predictor with subsequent characteristics:
   i. uses gradient descent method with learning rate of 0.5
   ii. uses mean square error loss
   iii. performs 1000 iterations which update your parameters using gradient descent method

(c) Compare your results using the gradient descent approach with the analytical one you have implemented in `find_theta` function in the previous homework. Are they the same? If not, what could be the reasons for that? Tune your hyperparameters (number of iterations and learning rate) to improve your performance.
(d) Plot your samples and your predictor. Hint: Take a look at Tracker.data().

3. Introduction to classification. You are given a dataset boolean_classification.json with raw data input $x_i^T \in \mathbb{R}^d$ represented in a $n \times d$ matrix $X$ and classes $y_i \in \mathbb{R}$ represented in a $n$-vector $y$. This time you are explicitly told that $y_i$ is discreet-valued and represents the class of the sample. Train your model for the entire dataset.

(a) Plot the dataset and highlight two different classes with different colors.

(b) Perform the least squares regression using the embedding $x = (1, u)$ and $y = v$ and return the predicted class for the given dataset. How would you predict the class using this predictor?

(c) Plot the samples and your prediction regions for each of the class.

(d) Repeat part (b) and (c) for the nearest-neighbor predictor. Comment the results. How would you implement the k-nearest neighbor predictor? Would it perform better?